REMARKS

Applicant wishes to thank the Examiner for his comments in the Advisory Action. Claims 1 and 7 have been amended consistent with the Examiner's comments.

Claims 1 and 3-25 are currently active.

Claim 2 has been canceled.

The Examiner has indicated that Claims 5, 6 and 10-16 are allowable if rewritten in independent form with all the limitations of their base claim and any intervening claims. Claim 23 is Claims 2-5 written as such, and Claim 17 is Claims 7-10 written as such. The Examiner has allowed Claims 17-25.

The Examiner has objected to the drawings. Substitute and corrected drawings are included to obviate this objection.

The Examiner has rejected Claims 2-4 and 7 as being anticipated by Reuter1. Claim 2 has been canceled. Claims 3 and 4 now depend on Claim 1. Claim 7 has been

amended to include the limitation of Claim 1 regarding the timestamp-based voting algorithm and as amended per the Examiner's comments in the Advisory Action.

The Examiner has rejected Claims 1, 8 and 9 as being unpatentable over Reuter1 in view of Reuter2. Applicant respectfully traverses this rejection in view of the amendment to the claims per the Examiner's comments in the Advisory Action.

Referring to Reuter1, there is disclosed a system and method for effectuating distributed consensus among members of a processor set in a multiprocessor computing system through the use of shared storage resources. Reuter1 teaches, with reference to figure 2, competing processors 202, 204, 206, and 208 represent processors having coordinated access to shared storage location 222. Processor 202 is associated with processor identifier 214, that uniquely identifies the processor from all other processors having access to the shared storage location 222. Likewise, processor 204 is associated with ID 216, processor 206 is associated with ID 218, processor 208 is associated with ID 220. See column 5, line 55-column 6, line 3.

Reuter1 teaches the shared storage location comprises a single unit of storage, a block of storage units, and multiple blocks of storage locally or remotely stored within the system. Reuter1 specifically teaches that the shared storage location comprises a magnetic

disk drive. The shared storage location is associated with two critical storage blocks. The critical storage block is a data structure such as a register, table, or other memory or storage device. The shared storage location 222 is associated with the critical storage block 1 (CS1) 242 and the critical storage block 2 (CS2) 244. The CS1 242 preferably comprises processor identifier field 226 and current counter field 228. The critical storage blocks provide not only means for storing state information to reserve for a mutual exclusion lock, but also a processor id field that stores the processor id. The processor id field enables a processor to detect not only the critical storage block is reserved, but identifies a specific processor that has made the reservation. See column 6, lines 16-40.

The critical storage block includes a counter field that stores a counter value to provide correctness in cases where a processor crashes and restarts the process. The counter value is incremented each time a specific processor reserves the critical storage block. The CS1 and CS2 sections may be read from and written to by any processor in a shared set of competing processors according to a mutual exclusion protocol. See column 6, lines 49-66 of Reuter1. By coordinating accesses to the CS1 242 and the CS2 244, a processor can establish an exclusive access reservation that can be detected and honored by other competing processors, without data corruption or unacceptable delays relating to race conditions or processor crashes. The programmable storage location 250 stores a race wait delay value, and storage allocation 252 stores a crash detect delay value. Each value is a time value or a clock

cycle count. Each processor is associated with the same place settings as every other processor that enters the set of competing processors, although delay settings for individual processes may be tuned.

As is evident from the above, Reuter1 fails to teach or suggest the added limitation of using a timebased-stamp voting algorithm, as found in amended Claim 1. Not only does Reuter1 fail to teach or suggest this data limitation, but Reuter1 fails to even recognize the problem and the advantage that this limitation added to the claims solves. The Examiner recognizes this and cites Reuter1 in regard to this limitation.

Referring to Reuter2, there is disclosed a system and method for exclusive access to shared storage. Reuter2 teaches that the system and method provides safe behavior when the storage subsystem does not behave with expected read and write times. The only time constraint that is preferably honored in an underlying disk subsystem is a "crash detect delay" to determine when a processor has crashed when holding or attempting to reserve exclusive access. See column 3, lines 28-37.

Reuter2 teaches the system and method implements a mutex lock that enables processes and/or processors competing for a shared resource to detect and resolve race conditions without undue overhead. The system and method also enables competing processors to detect a

crash condition that is causing one processor to refuse to release a lock on the shared resources. The crash detection feature allows an incoming processor to seize control of the lock when the crash is detected to prevent a crash induced deadlock. Another important feature of the crash detect feature is that it allows a processor that has crashed to restart and resume control of the lock in a manner that prevents other processors from prematurely pre-empting the restarting processor. See column 4, line 59-column 5, lines 7.

Reuter2 teaches there are two critical storage blocks CS1 246 and CS2 248. The critical storage blocks provide not only the means for storing state information to reserve the lock, but also a processor id field that stores a unique processor id. Processor id field enables any processor to detect not only that the critical storage block is reserved, but identifies a specific processor that has made the reservation. It is simply a predefined reserve code that is not associated with a processor and a set of competing processors. The critical storage block includes a counter field that stores a counter value to provide correctness in cases were a processor crashes and restarts the process. The counter value is incremented each time a specific processor reserves the critical storage blocks. If that processor crashes and upon restarting attempts to reserve a critical storage block again, the counter is incremented. In this manner, other processors will know that even though the first processor has held a lock for a time period longer than the crash detect delay, that it has to be started. This prevents the other processors

from seizing control of the lock and storing their own processor id in the processor id field while the first processor is in fact using the shared resources. See column 6, lines 29-61.

The two critical storage blocks may be read from and written to by any processor and a shared set of competing processors. By coordinating access to the two critical storage blocks, a processor can establish an exclusive access reservation that can be detected and honored by other competing processors, without data corruption or unacceptable delays related to race conditions or processor crashes. See column 6, line 65-column 7, lines 6.

Reuter2 teaches preferable storage location 250 stores a "Race Weight Delay" value, and storage location 252 stores a crash detect delay value. Each value is a time value of a clock cycle count. The race weight delay is used by a processor to cause a processor to wait long enough to allow another processor in the system to read and write to a critical storage block. The first processor writes processor id into one critical storage block, and if the contents in it are unchanged after the race weight delay, the processor is assured that another processor has not intervened when the first processor attempt to reserve the critical block (race condition does not exist). The crash detect delay value is used by a processor to determine that another processor that had previously reserved or attempted to reserve the shared storage location but has crashed, thereby making the shared storage location available to other ones of the competing processors. The crash detect delay value is long in comparison to the race weight delay. Because crashes are

rare, access requests will seldom incur the crash detect delay and it is of less concern. Crash detect delay is used to recover from the unlikely circumstance where processor read/write times exceed the race weight delay. See column 7, lines 7-51.

As is evident from the above description of Reuter2, there is no teaching or suggestion whatsoever of the limitation of a time stamp-based voting algorithm for the primary server to arbitrate access of the servers to the set of disks. The algorithm taught by Reuter2 is for the purpose of preventing crash-induced deadlock.

Claims 3 and 4 are dependent to parent Claim 1 and are patentable for the reasons Claim 1 is patentable.

In view of the foregoing amendments and remarks, it is respectfully requested that the outstanding rejections and objections to this application be reconsidered and withdrawn, and Claims 1 and 3-25, now in this application be allowed.

Respectfully submitted,

MICHAEL LEON KAZAR

By ______________________

Ansel M. Schwartz, Esquire
Reg. No. 30,587
One Sterling Plaza
201 N. Craig Street
Suite 304
Pittsburgh, PA 15213
(412) 621-9222

Attorney for Applicant

ANNOTATED MARKED UP DRAWING

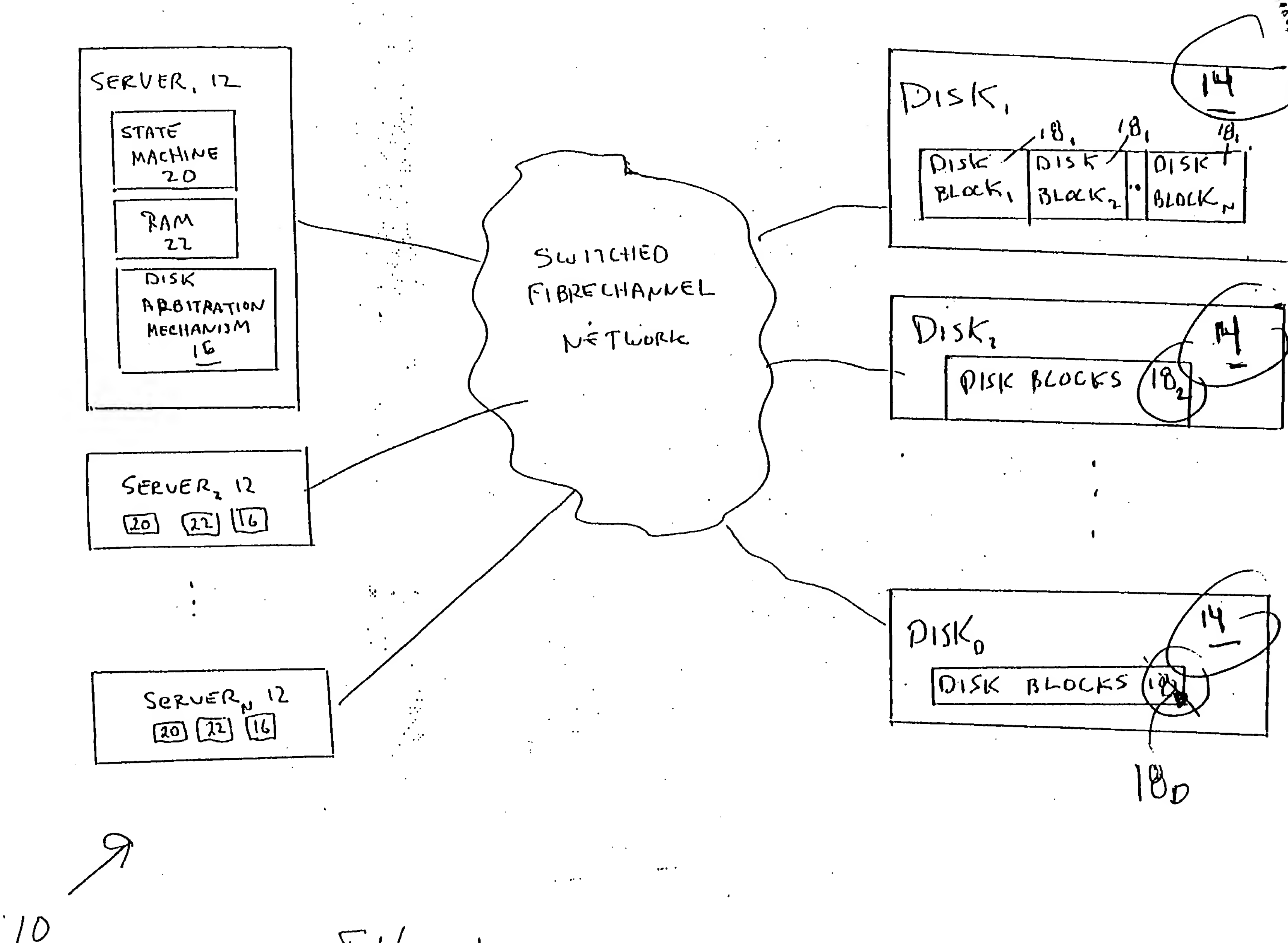

FIG 1

REPLACEMENT SHEET

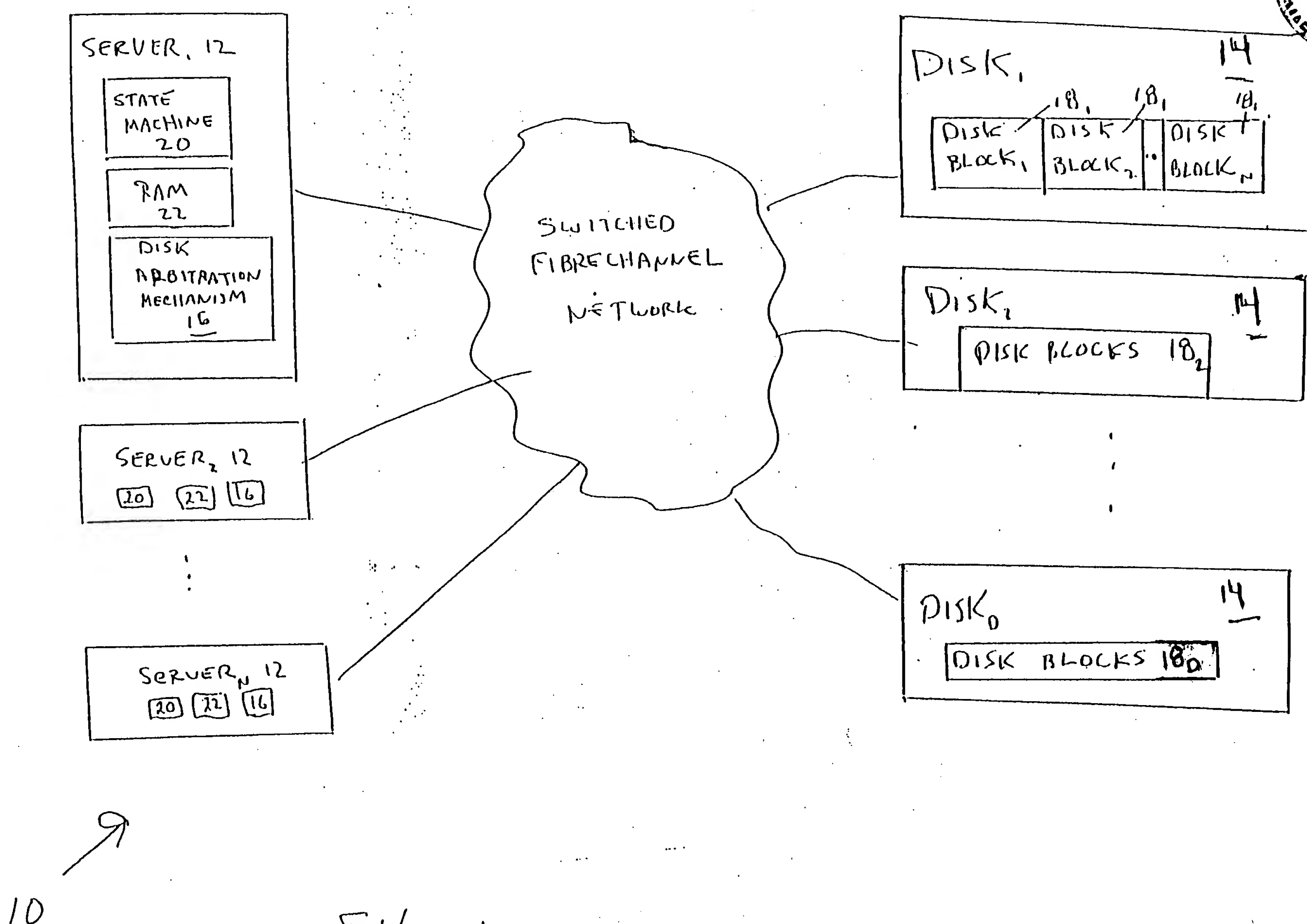

FIG 1